



DELTA: Dynamic Layer-Aware Token Attention for Efficient Long-Context Reasoning

Hossein Entezari Zarch, Lei Gao, Chaoyi Jiang, Murali Annavaarm

University of Southern California

{entezari, leig, chaoyij, annavara}@usc.edu

Abstract

Large reasoning models (LRMs) achieve state-of-the-art performance on challenging benchmarks by generating long chains of intermediate steps, but their inference cost is dominated by decoding, where each new token must attend to the entire growing sequence. Existing sparse attention methods reduce computation by pruning the key-value (KV) cache, yet they suffer from severe accuracy degradation on reasoning tasks due to cumulative selection errors and the dynamic importance of tokens over long derivations. We present **DELTA**, a training-free sparse attention mechanism that achieves computational efficiency without sacrificing model accuracy. DELTA partitions transformer layers into three groups: initial layers that use full attention, a small set of *selection layers* that identify salient tokens via aggregated head-level attention scores, and subsequent *sparse-attention layers* that attend only to the selected subset. This design preserves the full KV cache in GPU memory for accuracy, while avoiding expensive full-attention computation over many layers. On reasoning benchmarks such as AIME and GPQA-Diamond, DELTA matches or surpasses full attention in accuracy, while reducing the number of attended tokens by up to 5 \times and delivering 1.5 \times end-to-end speedup. Our results show that selective reuse of intermediate attention maps offers a robust path toward efficient long-context reasoning.

1 Introduction

Recent progress in large language models (LLMs) has led to systems with impressive capabilities in reasoning and self-reflection. Large reasoning models (LRMs) such as DeepSeek-R1 (Guo et al., 2025), Gemini-2.5-pro (Google DeepMind, 2025), OpenAI-o3 (OpenAI, 2025b), Qwen3 (Yang et al., 2025), and GPT-OSS (OpenAI, 2025a) leverage test-time scaling by generating long chains of intermediate reasoning steps, significantly improving accuracy on challenging benchmarks (AIME, 2025;

Rein et al., 2024; Hendrycks et al., 2021; Wei et al., 2022). However, serving such models efficiently remains difficult due to severe memory and compute bottlenecks in attention operation (Vaswani et al., 2017), especially under long-context generation settings (Dao, 2023; Ye et al., 2025).

LLM inference consists of two stages: *prefilling* and *decoding*. In the prefilling stage, the model processes the prompt, computes hidden representations, and materializes all key–value (KV) vectors as a KV cache in GPU high-bandwidth memory (HBM). During decoding, tokens are generated autoregressively: for each new token, the model computes its KV vectors, appends them to the cache in HBM, and attends over the entire history to produce the next output. Because the KV cache grows linearly with sequence length and batch size (Kwon et al., 2023), the amount of data that must be read from HBM increases rapidly. For instance, with a 32K-token context and a batch size of 128, the KV cache of LLaMA-3-8B in float16 already exceeds 500 GB.¹ Unlike the prefilling stage, which writes the KV cache once, the decoding stage must repeatedly stream all previously stored KV entries from HBM for every new token. This makes decoding inherently *memory-bandwidth bound*: throughput is limited by the cost of moving hundreds of gigabytes of KV data per step. As context length or batch size grows, this bandwidth pressure scales linearly, quickly overwhelming GPU memory systems and severely constraining long-context inference.

These bandwidth limitations are particularly acute for reasoning workloads. Unlike typical NLP tasks that involve long inputs but short outputs, reasoning problems often begin with concise prompts yet require lengthy derivations spanning tens of thousands of tokens. This decode-heavy profile magnifies the bandwidth bottleneck, as each step

¹Computed as Layers \times Sequence Length \times Batch Size \times KV Heads \times Head Dim \times 2 (for K&V) \times 2 bytes = 32 \times 32K \times 128 \times 8 \times 128 \times 2 \approx 512 GB.

involves scanning ever-larger KV caches. As a result, decoding stage dominates both latency and resource usage: for example, using full attention in HuggingFace, DeepSeek-R1-Distill-Llama-8B requires more than 15 minutes on a single NVIDIA A100 GPU to generate 32K tokens for one AIME problem (Yue et al., 2025). Optimizing the decoding stage is therefore essential for efficient LLM serving in reasoning applications.

Meanwhile, the unique structure of reasoning workloads opens new opportunities for efficiency. While prefilling benefits from full attention to capture global context, the much longer decoding phase is well-suited to sparsity. Sparse attention reduces computation and bandwidth requirements by restricting reads to a subset of salient tokens rather than scanning the entire KV cache. Prior work has explored two complementary directions: selection-based methods (Tang et al., 2024; Hao et al., 2025; Liu et al., 2024a; Gao et al., 2025; Yuan et al., 2025; Yang et al., 2024), which preserve the full KV cache but attend only to chosen tokens, and eviction-based methods (Hu et al., 2025; Li et al., 2024; Xiao et al., 2023b; Zhang et al., 2023; Adnan et al., 2024; Cai et al., 2025), which permanently discard unselected tokens to reduce storage cost of KV cache. Both rely on identifying important tokens using predefined criteria, and together they demonstrate the potential of sparse attention as a foundation for efficient long-decode inference.

However, applying sparse attention to long reasoning generations remains challenging. Unlike standard generation tasks, where some information loss can be tolerated, step-by-step reasoning demands that critical context be preserved throughout the entire derivation to maintain logical consistency (Hu et al., 2025). In practice, accuracy drops sharply when token selection errors accumulate over long sequences (Gao et al., 2025). Eviction-based methods such as RaaS (Hu et al., 2025) illustrate this issue: by permanently removing tokens judged less important, they risk discarding tokens that later become essential once the generation length grows beyond the KV cache capacity. The core difficulty is twofold: (1) attention patterns evolve over time, and (2) a token’s importance can change, tokens that seem irrelevant early may become highly influential later in the reasoning process.

At the same time, we make two key observations. First, attention maps across consecutive layers exhibit strong correlation: within a local block of

layers, the first layer often predicts the important tokens for subsequent layers with high reliability. Second, attention distributions change gradually during decoding, which suggests that token importance can be predicted using intermediate layers without computing full attention everywhere. These insights highlight both the risk of aggressive eviction and the opportunity for accurate, low-cost selection.

To address the accuracy–efficiency tradeoff, we introduce **DELTA**, a training-free, selection-based sparse attention mechanism. DELTA preserves the full KV cache but restricts computation to a carefully chosen subset of tokens at each decoding step. It operates as a plug-and-play module that leverages the full attention maps of a small set of intermediate layers to predict the salient tokens for the upcoming layers. By accurately identifying high-attention tokens while maintaining a stable recency window, DELTA significantly reduces the runtime cost of attention without incurring noticeable accuracy degradation.

In summary, our contributions are:

- We provide a detailed token-level analysis of attention distributions in large reasoning models, revealing two properties: (1) strong correlation of attention patterns across consecutive layers, and (2) gradual but ongoing shifts in token importance during long generations.
- We propose **DELTA**, a training-free sparse attention mechanism that combines (a) *Unified Head Selection*, which globally aggregates head-level top- k tokens, with (b) a *Stable Recency Window*, which guarantees retention of recent context crucial for reasoning.
- We demonstrate that DELTA achieves accuracy on par with, or better than, full attention on challenging reasoning benchmarks, while delivering up to $1.5\times$ end-to-end speedups. Compared with state-of-the-art sparse attention methods, DELTA reduces the number of attended tokens by up to $5\times$, all without sacrificing accuracy.

2 Background

LLM inference process. Decoder-only LLMs generate tokens auto-regressively in two stages: the prefilling stage and the decoding stage.

Prefilling stage. At layer i , the input hidden states are $X^i \in \mathbb{R}^{b \times s \times h}$, where b is the batch size, s is the

prompt length, and h is the embedding dimension. Queries, keys, and values are projected as

$$Q^i = X^i W_Q^i; K^i = X^i W_K^i; V^i = X^i W_V^i \quad (1)$$

where

$$W_Q^i \in \mathbb{R}^{h \times h}, \quad W_K^i, W_V^i \in \mathbb{R}^{h \times (g \cdot d_{\text{head}})}.$$

Here m denotes the number of query heads, $g \leq m$ the number of KV groups, and $d_{\text{head}} = h/m$ the per-head dimension.

In grouped-query attention (GQA), the queries are divided into m query heads,

$$Q^i = [Q_1^i, \dots, Q_m^i], \quad Q_j^i \in \mathbb{R}^{b \times s \times d_{\text{head}}}, \quad (2)$$

while the keys and values are divided into only g groups,

$$\begin{aligned} K^i &= [K_1^i, \dots, K_g^i], \quad V^i = [V_1^i, \dots, V_g^i], \\ K_\ell^i, V_\ell^i &\in \mathbb{R}^{b \times s \times d_{\text{head}}}. \end{aligned} \quad (3)$$

Each query head j is assigned to one KV group $\phi(j) \in \{1, \dots, g\}$. GQA generalizes standard attention mechanisms: when $g = m$, it reduces to multi-head attention (MHA), and when $g = 1$, it reduces to multi-query attention (MQA).

The scaled dot-product attention for head j is

$$A_j^i = \frac{Q_j^i (K_{\phi(j)}^i)^\top}{\sqrt{d_{\text{head}}}}, \quad O_j^i = \text{softmax}(A_j^i) V_{\phi(j)}^i, \quad (4)$$

where A_j^i are the attention scores and $O_j^i \in \mathbb{R}^{b \times s \times d_{\text{head}}}$ is the head output.

The outputs of all query heads are concatenated and linearly projected:

$$O^i = [O_1^i, \dots, O_m^i] W_O^i, \quad W_O^i \in \mathbb{R}^{h \times h}. \quad (5)$$

A feed-forward network (FFN) follows the GQA block:

$$X^{i+1} = \sigma(O^i W_1^i) W_2^i, \quad (6)$$

where $W_1^i \in \mathbb{R}^{h \times d_{\text{FFN}}}$, $W_2^i \in \mathbb{R}^{d_{\text{FFN}} \times h}$, and $\sigma(\cdot)$ is a non-linear activation.

Decoding stage. At step t , each layer receives a single token embedding $x^i \in \mathbb{R}^{b \times 1 \times h}$. The new key and value are concatenated to the cached ones:

$$K^i \leftarrow [K^i; x^i W_K^i], \quad V^i \leftarrow [V^i; x^i W_V^i]. \quad (7)$$

The subsequent GQA and FFN computations mirror the prefilling stage.

Decode cost and memory I/O. While prefilling writes the KV cache once, decoding must repeatedly read all past K/V entries for each new token, making long-context inference inherently memory-bandwidth bound. Prior work reports that decoding dominates end-to-end latency under long contexts and that KV memory movement constitutes a major fraction of decode time, underscoring the need to reduce KV reads without sacrificing accuracy (Kwon et al., 2023; Dao, 2023).

Sparse attention. Full attention requires each query to attend to all past tokens, which scales linearly with sequence length. Sparse attention reduces this cost by restricting computation to a subset of k tokens. For head j , let the exact attention weights be

$$\alpha_j^i = \text{softmax}(A_j^i) \in \mathbb{R}^s.$$

Instead of attending to all s tokens, we select an index set $\rho \subseteq \{1, \dots, s\}$ with $|\rho| = k$. Since computing ρ from α_j^i directly is expensive, practical methods rely on an approximation function f that predicts which tokens are likely to have high attention:

$$\rho = \arg \max_{\rho': |\rho'|=k} f(Q_j^i, K_{\phi(j)}^i, V_{\phi(j)}^i, \rho'). \quad (8)$$

The quality of the selection is measured by the *attention recall*, defined as the fraction of the ground-truth attention mass preserved in the selected subset:

$$R_j^i = \frac{\sum_{u \in \rho} \alpha_j^i(u)}{\sum_{u=1}^s \alpha_j^i(u)}. \quad (9)$$

Maximizing R_j^i under the budget constraint k is the central objective of sparse attention methods, ensuring efficiency while maintaining accuracy.

Sparsity and query dependence. Self-attention exhibits substantial sparsity beyond the earliest layers: a small subset of critical tokens typically accumulates most attention mass, enabling accurate computation on a reduced context. However, criticality is strongly *query dependent*: the tokens that matter vary with the current query vector Q , and may change rapidly across consecutive decode steps. Heuristics based only on past usage (eviction) risk losing later-salient tokens, whereas query-aware selection retains high recall under long reasoning traces (Tang et al., 2024; Zhang et al., 2023; Ge et al., 2023).

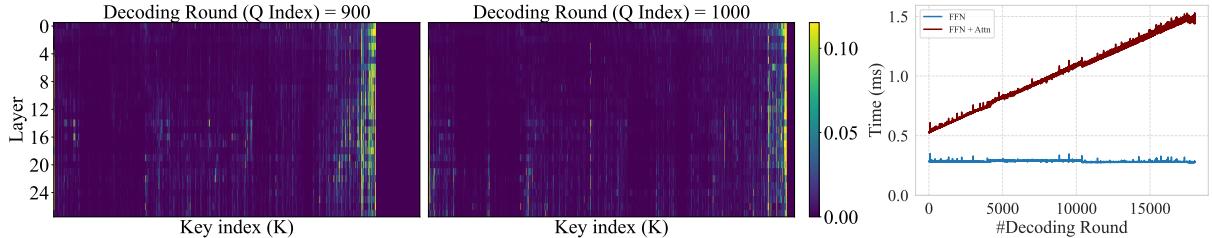


Figure 1: (Left) Attention maps from Qwen-7B at decoding steps 900 and 1000, where each row corresponds to a Transformer layer. (Right) Decoding runtime of FFN and attention modules across generation, showing attention’s linear growth with context length.

3 Motivating Observation

Depth-wise context sharpening. Figure 1 (left) illustrates how attention patterns evolve with depth. In early layers, the model primarily attends to nearby tokens and exhibits diffuse, low-mass attention over the broader context, showing little focus on distant information. As depth increases, attention becomes progressively sharper and more selective, concentrating on a small set of far-away tokens that carry high relevance.

Layer-wise correlation. Empirical profiling of large reasoning models such as Qwen-7B reveals that consecutive layers exhibit highly correlated attention patterns. Tokens that receive high attention in one layer tend to remain salient in the next layers, as illustrated in Figure 1 (left), which visualizes layer-wise attention maps at decoding steps 900 and 1000 of a reasoning sequence. Each row corresponds to a Transformer layer, showing that deeper layers largely preserve the spatial configuration of attention established in earlier layers. This structural continuity suggests that adjacent layers refine rather than reconstruct attention, enabling later layers to reuse the relational patterns captured by their predecessors. As a result, computing full attention in every layer becomes redundant: once salient tokens are identified, subsequent layers can effectively operate on a reduced, high-recall subset of the context.

Sequential drift. While the overall structure of attention is stable across depth, it evolves gradually along the decoding trajectory. Between decoding steps 900 and 1000 in Figure 1, the regions of strongest attention shift across key positions, revealing how the model dynamically repositions its focus as new tokens are generated. We refer to this phenomenon as *sequential drift*.

This progressive movement reflects an adaptive retrieval process, where the model continuously

updates which parts of the context are relevant to the current query embedding Q_t . Such behavior highlights the need for a *query-adaptive* sparse attention mechanism that dynamically adjusts token selection at each decoding step rather than relying on fixed or history-based heuristics.

Runtime dominated by attention. Figure 1 (right) presents the measured decoding runtimes of FFN and attention modules. While the FFN cost remains nearly constant, attention latency increases almost linearly with context length. Beyond 8k tokens, attention dominates total inference time, driven primarily by repeated KV-cache memory access rather than compute operations. These trends confirm that long-context decoding is bottlenecked by attention and motivate our approach: performing full attention only in a few strategically chosen layers to identify salient tokens, while letting the remaining layers operate on a compact, high-recall context subset.

4 DELTA: Dynamic Layer-Aware Token Attention

The empirical patterns described Section 3 motivate a layer-aware sparse attention design that minimizes redundant computation while preserving reasoning accuracy. In early layers, attention maps are diffuse and unstable, requiring full-sequence attention to build reliable representations. In contrast, deeper layers exhibit high inter-layer correlation: once a small set of context tokens becomes salient, subsequent layers largely reuse them. Finally, as decoding proceeds, the regions of strong attention shift gradually along the sequence, a phenomenon we term *sequential drift*. Together, these observations suggest that only a few layers need to compute full attention to refresh salient tokens, while the rest can reuse them at low cost.

Core idea. DELTA operationalizes this principle through a structured three-tier layer design. The

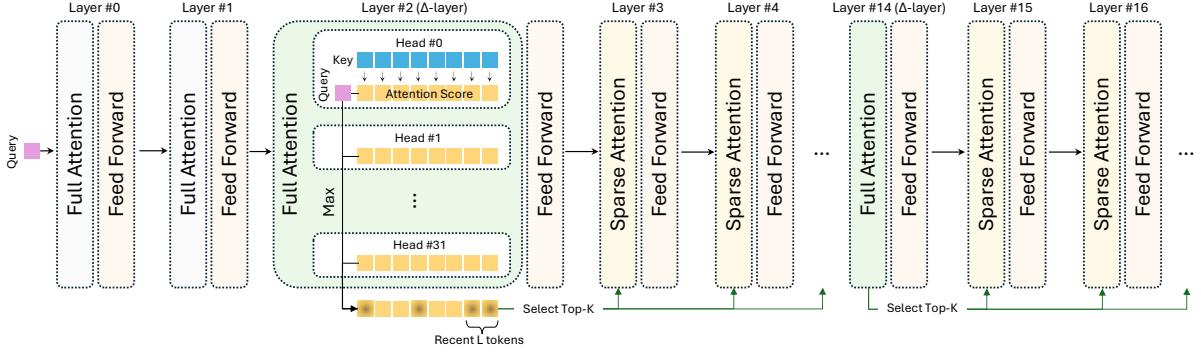


Figure 2: Overview of the DELTA decoding process. The first two layers perform full attention for initialization, Δ -layers (e.g., Layers 2 and 14) run full attention to select salient tokens, and subsequent sparse attention layers attend only to those selected tokens, as indicated by green arrows.

first few layers perform full attention to stabilize representations, as early layers show no consistent sparsity structure. A small number of intermediate Δ -layers act as *selection layers* that re-run full attention to identify a compact set of salient tokens carrying most of the attention mass. The remaining layers perform sparse attention restricted to those selected tokens, reusing them until the next Δ -layer updates the selection. This design removes redundant full-sequence computation across correlated layers while maintaining a high-recall context for reasoning.

Query-adaptive refresh. Because attention focus evolves with each new query embedding Q_t , the salient set must be updated at every decoding step. Skipping or caching old selections would cause stale focus and recall loss. Therefore, each Δ -layer recomputes full attention per generated token, ensuring that the reduced context remains aligned with the evolving query. However, this update occurs only at the sparse set of Δ -layers, keeping total computation cost low. Notably, DELTA never discards tokens from the KV cache: older tokens may regain relevance due to long-range reasoning dependencies. Instead, it restricts computation, not memory, by letting sparse attention layers attend only to the currently selected subset while retaining the full cache for later refreshes.

Selection mechanism. At each Δ -layer i , we form a reduced context ρ by preserving a small recency window and selecting older tokens with the highest importance. Let $\alpha_j^i = \text{softmax}(A_j^i) \in \mathbb{R}^{b \times s}$ denote the head-wise attention weights (Eq. 4). The importance of token t is defined as its maximum attention value across heads:

$$s_t = \max_{j=1,\dots,m} \alpha_j^i(\cdot, t).$$

We then select the top- $(k - L)$ tokens by s_t among $\{1, \dots, s - L\}$ and merge them with the recency window:

$$\rho = \text{Topk}(\{s_t : t \leq s - L\}, k - L) \cup \{s - L + 1, \dots, s\}.$$

Page-based DELTA. Token-level KV management fragments memory and hinders efficient GPU access. Following common practice, we store the KV cache in fixed-size pages of P tokens (Kwon et al., 2023; Dao, 2023). Let $\mathcal{P} = \{1, \dots, \lceil s/P \rceil\}$ be the page set and $p(t) \in \mathcal{P}$ map token t to its page. We define the page score S_u as the sum of its token scores:

$$S_u = \sum_{t: p(t)=u} s_t.$$

To preserve recency at page granularity, we always keep the last L/P pages (covering the most recent L tokens), and among the remaining pages we select the top- $k - (L/P)$ by S_u . The reduced context is the union of tokens within the preserved and selected pages. This block structure enables coalesced memory access and reduces gather/scatter overhead during decoding.

5 Experiments

Experimental setup. We evaluate DELTA on three distilled variants of DeepSeek-R1 (Guo et al., 2025): DeepSeek-R1-Distill-Qwen-1.5B, 7B, and 14B, denoted as Qwen-1.5B, 7B, and 14B for brevity. The models’ reasoning performance is assessed on four open-source mathematical benchmarks: AIME-2024, AIME-2025 (AIME, 2025), GPQA-Diamond (Rein et al., 2024), and MATH500 (Hendrycks et al., 2021). For each benchmark, we evaluate on 30 test cases: all 30

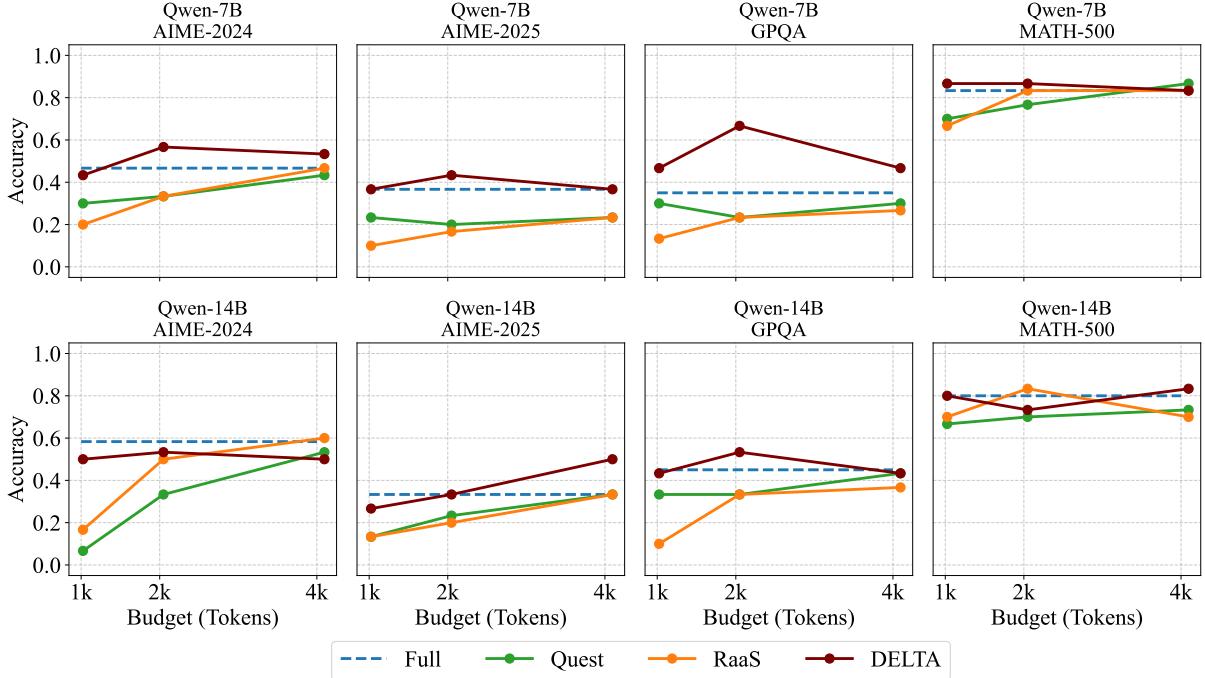


Figure 3: Accuracy of sparse attention methods on reasoning benchmarks using Qwen-7B and 14B models. DELTA consistently matches or exceeds the accuracy of Full attention under limited token budgets and maintains robustness across different datasets.

problems in AIME-2024/2025 and the first 30 problems in GPQA-Diamond and MATH500.

Datasets. The AIME datasets contain problems from the American Invitational Mathematics Examination, which challenge top U.S. high school students with advanced questions in algebra, geometry, number theory, and combinatorics. MATH500 is drawn from high school competitions and organized into five difficulty levels following the Art of Problem Solving (AoPS) framework, ranging from introductory to Olympiad-level. GPQA-Diamond is a subset of the Graduate-level Google-Proof Question Answering benchmark, designed to test deep multi-step reasoning in science and mathematics tasks where surface-level pattern matching is insufficient.

Implementation details. We implement DELTA using the FlashInfer Just-In-Time (JIT) module (Ye et al., 2025) to extract attention scores on-the-fly from the decoding kernel, and apply the native PyTorch topk operator for token selection. This design avoids custom CUDA kernels, making DELTA lightweight and easy to integrate across models. For all settings, we employ paged KV caching with a page size of $P = 16$, following common practice (Kwon et al., 2023; Dao, 2023). Baseline implementations (Full, Quest, etc.) are reproduced in

Hugging Face Transformers (Hugging Face, 2025) for consistent comparison. All experiments are conducted on a single NVIDIA A100 (SXM4, 40GB) GPU with CUDA 12.8 and 16 CPU cores from an AMD EPYC 7H12 processor.

Layer configuration. For both Qwen-7B and Qwen-14B, layers $[0, 1]$ are regular full attention for initialization. Qwen-7B employs layers $[2, 14, 22]$ (out of $[0-27]$) as Δ -layers, while Qwen-14B uses layers $[2, 6, 42]$ (out of $[0-47]$). These layers are selected based on a calibration study of attention-map dynamics: they correspond to points showing the largest average shift in attention distributions across decoding steps, marking transitions where salient-token sets change most rapidly.

Baselines. We compare DELTA against three approaches. **Full** denotes standard decoding where all layers attend to the entire KV cache. **Quest** (Tang et al., 2024) is a selection-based method that compresses each KV page into two representative vectors (element-wise min/max of keys), scores pages against the current query, and retrieves the top- k for attention; it preserves the full cache in HBM but incurs overhead from storing representatives (two key vectors per page, i.e., 1/8 of KV memory for a page size of 16). **RaaS** (Hu et al.,

2025) is an eviction-based method that removes pages with consistently low attention scores, lowering memory usage but risking the loss of tokens that may later become important.

Metrics. The first metric is accuracy, measuring whether the model’s final answer is mathematically equivalent to the ground truth. Each test case is marked as either correct or incorrect, and overall accuracy is reported as the percentage of correctly solved problems across the dataset. The second metric is the number of decoding steps, which reflects how many tokens the model generated before reaching the end-of-sequence token or the maximum generation limit.

5.1 Accuracy Under Varying Token Budgets

Figure 3 compares the accuracy of three sparse-KV methods, Quest, RaaS, and DELTA, across four reasoning datasets and two model scales. Three consistent patterns emerge. First, with token budget (1K), DELTA consistently outperforms existing sparse methods and often matches or even surpasses the Full attention baseline. For instance, on AIME-2024 with Qwen-14B, Quest and RaaS achieve below 20% accuracy, whereas DELTA attains nearly 50%, approaching the 60% accuracy of Full attention. Second, increasing the token budget from 1K to 2K often improves performance, reflecting the diminishing effect of sparsity-induced selection errors. In several cases, DELTA with 2k token budget even surpasses the Full-KV baseline, for example, Qwen-7B on GPQA, where DELTA outperforms Full attention by roughly 30%. Finally, expanding the budget further to 4K yields marginal or no improvement and occasionally a slight decline in accuracy. This plateau suggests that DELTA captures most salient context within small budgets, beyond which additional tokens primarily introduce redundancy rather than useful information.

5.2 Speedup Results

Figure 4 (left) presents the cumulative distribution function (CDF) of generation lengths across evaluated samples. Lower curves indicate shorter generation lengths, which are desirable since longer outputs imply inefficiency or reasoning drift. As shown, DELTA consistently achieves lower or comparable generation lengths relative to Full attention, while outperforming other sparse-KV methods, confirming that its sparsity does not lengthen reasoning traces. Figure 4 (right) reports the per-round decoding latency when using the Full KV

cache versus DELTA with a 1K token budget. Experiments were conducted on Qwen-1.5B with a batch size of 64 and a maximum generation length of 18K tokens. The gray vertical line marks the point where DELTA’s token selection begins. Beyond this point, the growth rate of per-step latency with respect to context length notably decreases: while the Full-KV runtime rises from 27.5 ms to 35 ms, DELTA grows only from 27.5 ms to 28 ms. At a context length of 18K, each decoding round under DELTA takes about 28 ms compared to 35 ms for Full KV. Overall, the total generation time drops from 968 to 649 seconds, a 33% reduction in decoding latency, while throughput improves from 1,194 tokens/s to 1,774 tokens/s. Together, these results demonstrate that DELTA preserves generation efficiency without increasing output length, achieving substantial end-to-end speedups on real reasoning workloads.

5.3 Effect of Recency Window Size

Table 1 shows the effect of the recency window size L on accuracy for different total token budgets K using Qwen-7B on AIME-2025. When the budget is small (e.g., $K=64$), larger recency windows (e.g., $L=32$) yield higher accuracy, indicating that recent context is most valuable under tight memory. As K increases (e.g., 128–256), smaller L becomes optimal, allowing more capacity for long-range salient tokens selected by DELTA. This trend highlights a trade-off between recency and contextual breadth, where larger budgets favor wider attention coverage while smaller ones rely on preserving recent context.

K	L					
	1	2	4	8	16	32
64	0.27	0.30	0.30	0.37	0.40	0.43
128	0.40	0.43	0.43	0.47	0.43	0.40
256	0.37	0.50	0.40	0.37	0.37	0.40

Table 1: Accuracy across different token budget K and recency window L for Qwen-7B on AIME-2025. Larger K tend to yield higher accuracy with smaller L .

6 Related Work

Efficient long-context inference. Long-context LLMs face quadratic compute and memory overhead from full self-attention, making inference increasingly dominated by KV-cache bandwidth. Even with optimized kernels such as FlashAttention (Dao, 2023) and paged caching (Kwon et al.,

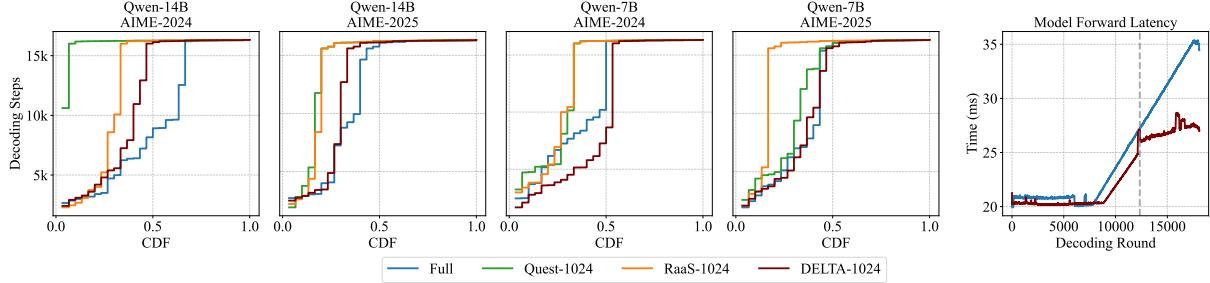


Figure 4: (Left) CDF of decoding rounds across model-dataset pairs. DELTA reaches high CDF values faster than baselines by maintaining shorter generation lengths. (Right) End-to-end forward latency per decoding round. After DELTA activation (gray line), latency becomes lower than full attention.

2023), decoding throughput scales poorly with sequence length. Modern architectures (e.g., LLaMA-3.1, GPT-4o, Claude 3.5 Sonnet) extend context to 128K–200K tokens through rotary positional encoding (Su et al., 2024), yet runtime remains bottlenecked by repeated KV reads rather than arithmetic compute, highlighting the need for structural sparsity that reduces redundant memory access.

Architectural and KV-compression methods. Architectural approaches such as Multi-Query and Grouped-Query Attention (Shazeer, 2019; Ainslie et al., 2023) reduce redundant KV heads, while recurrent alternatives like RWKV (Peng et al., 2023), RetNet (Sun et al., 2023), and Mamba (Gu and Dao, 2023) replace self-attention with stateful recurrence. These designs improve efficiency but require model retraining and often underperform Transformers on complex reasoning tasks. In contrast, KV-compression strategies optimize inference at runtime. Quantization (Xiao et al., 2023a; Yao et al., 2022; Dettmers et al., 2022; Liu et al., 2024b) lowers precision to save bandwidth, whereas pruning methods exploit sparsity to drop less important tokens. Eviction-based schemes such as H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), TOVA (Oren et al., 2024), ScissorHands (Liu et al., 2023), R-KV (Cai et al., 2025), and RaaS (Hu et al., 2025) bound memory by discarding low-score pages, but may lose tokens that later become critical for reasoning continuity.

Sparse and selection-based attention. Selection-based methods preserve the full KV cache but compute attention only over a subset of salient tokens. Early static patterns in Sparse Transformers, Longformer, and BigBird (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020) established fixed sparsity layouts, later refined into adaptive mechanisms guided by query-dependent importance. Quest (Tang et al., 2024) scores KV

pages against the current query to retrieve the most relevant subset, while TidalDecode (Yang et al., 2024) exploits the strong spatial coherence of attention across layers by performing full attention only in a few token-selection layers and reusing the selected tokens in intermediate sparse layers. SeerAttention-R (Gao et al., 2025) employs a self-distilled gating module to learn block-sparse attention, achieving near-lossless decoding but requiring additional training. However, existing sparse attention methods either incur notable accuracy degradation at low retention ratios or depend on costly post-training procedures to recover performance, both of which substantially increase decoding length and computational overhead for reasoning tasks. In contrast, **DELTA** is proposed as a selection-based, training-free approach that leverages inter-layer attention correlation during reasoning to maintain high accuracy under reduced token budgets, without extending the overall generation length.

7 Conclusion

We introduced DELTA, a training-free, layer-aware sparse attention mechanism that improves the efficiency of long-context reasoning in large language models. By leveraging cross-layer correlation and gradual evolution of token importance, DELTA computes full attention only in a few key Δ -layers and reuses their selected high-recall subsets across subsequent sparse attention layers. This design substantially reduces decoding-time bandwidth and latency while maintaining accuracy comparable to full attention. Experiments on multiple reasoning benchmarks confirm that DELTA achieves consistent speedups over state-of-the-art sparse and eviction-based methods without retraining, highlighting layer-aware reuse as a promising direction for efficient reasoning-time inference.

Limitations

While DELTA enables efficient long-context reasoning, it has the following limitations.

KV-memory footprint. DELTA preserves the full KV cache in HBM and reduces compute rather than peak memory. As a result, it does not directly address out-of-memory failures at extreme context lengths or on smaller GPUs. Future work includes integrating DELTA with complementary memory-saving techniques (e.g., quantization, eviction under guarantees, or offloading) while maintaining high selection recall.

Generality. Our results target distilled DeepSeek-R1 (Qwen-1.5B/7B/14B) on math reasoning. Transfer to other architectures, modalities, or conversational/code settings is unverified and may require re-tuning schedules and budgets.

Sensitivity and overhead. Performance depends on Δ -layer placement and (K, L) ; the max-attention scoring adds small overhead and can lag under fast attention drift. Adaptive per-sample scheduling or lightweight learned selectors are promising fixes.

References

- Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127.
- MAA AIME. 2025. American invitational mathematics examination-aime 2024, 2024.
- Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, and 1 others. 2025. R-kv: Redundancy-aware kv cache compression for training-free reasoning models acceleration. *arXiv preprint arXiv:2505.24133*.
- Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.
- Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332.
- Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun, Tianzhu Ye, Li Dong, and 1 others. 2025. Seerattention-r: Sparse attention adaptation for long reasoning. *arXiv preprint arXiv:2506.08889*.
- Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2023. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*.
- Google DeepMind. 2025. Gemini 2.5: Our most intelligent ai model. <https://blog.google/technology/google-deepmind/gemini-model-thinking-g-updates-march-2025/>. Accessed: 2025-09-27.
- Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shiron Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Jitai Hao, Yuke Zhu, Tian Wang, Jun Yu, Xin Xin, Bo Zheng, Zhaochun Ren, and Sheng Guo. 2025. Omnikv: Dynamic context selection for efficient long-context llms. In *The Thirteenth International Conference on Learning Representations*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.
- Junhao Hu, Wenrui Huang, Weidong Wang, Zhenwen Li, Tiancheng Hu, Zhixia Liu, Xusheng Chen, Tao Xie, and Yizhou Shan. 2025. Raas: Reasoning-aware attention sparsity for efficient llm reasoning. *arXiv preprint arXiv:2502.11147*.
- Hugging Face. 2025. Hugging face. <https://huggingface.co/>. Accessed: 2025-09-27.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Akyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.
- Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chen-gruidong Zhang, Bailu Ding, Kai Zhang, and 1 others. 2024a. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *arXiv preprint arXiv:2409.10516*.
- Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyridis, and Anshumali Shrivastava. 2023. Scisorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364.
- Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*.
- OpenAI. 2025a. Introducing GPT-OSS. <https://openai.com/index/introducing-gpt-oss/>. Accessed: 2025-09-27.
- OpenAI. 2025b. Introducing openai o3 and o4-mini. <https://openai.com/index/introducing-o3-and-o4-mini/>. Accessed: 2025-09-27.
- Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. 2024. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*.
- Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, and 1 others. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.
- Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.
- Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023a. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023b. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. 2024. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. *arXiv preprint arXiv:2410.05076*.
- Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183.
- Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yining Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and 1 others. 2025. Flashinfer: Efficient and customizable attention engine for llm inference serving. *arXiv preprint arXiv:2501.01005*.
- Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, and 1 others. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*.
- Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, and 1 others. 2025. Don't overthink it: A survey of efficient r1-style large reasoning models. *arXiv preprint arXiv:2508.02120*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuan-dong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.